

CLAIMS

We claim:

Sub P1
1. A multi-tasking operating system for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising:

at least one resource;

a plurality of threads; and

a stationary queue for allocating access to said resource amongst said threads.

2. A multi-tasking operating system stationary queue for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, the stationary queue comprising:

a sleep code routine for generating a unique block identifier when a process thread temporarily cannot gain access to said resource and must be suspended; and

a wakeup code routine for generating a unique run identifier when a next thread in line is to be re-animated and granted access to said resource.

1 3. The system of claim 2, further comprising:
 2 a wait counter for counting the cumulative number of
 3 threads that have been temporarily denied the resource;
 4 a satisfied counter for counting the cumulative number
 5 of threads that have been denied access and
 6 subsequently granted access to said resource;
 7 said sleep code routine being responsive to said wait
 8 counter for generating said run identifier; and
 9 said wakeup code routine being responsive to said
 10 satisfied counter for generating said run identifier.

1 4. A method for managing simultaneous access to scarce or
 2 serially re-usable resources by multiple process threads,
 3 comprising the steps of:
 4 responsive to a request for a resource which is not
 5 available, creating a block identifier based on the
 6 number of threads temporarily denied the resource; and
 7 blocking the thread using said block identifier.

1 6. A method for managing simultaneous access to scarce or
2 serially re-usable resources by multiple process threads,
3 comprising the steps of:

4 responsive to a request for a resource which is not
5 available,

6 creating a block identifier based on the number of
7 threads temporarily denied the resource; and

8 blocking the thread using said block identifier;
9 and

10 responsive to a resource becoming available,

11 creating a run identifier based on the number of
12 threads that have been first forced to wait and
13 have been subsequently satisfied; and

14 running the thread using said run identifier.

1 7. A memory device for storing signals for
2 controlling the operation of a computer to manage
3 simultaneous access to scarce or serially re-usable
4 resources by multiple process threads, according to the
5 steps of

6 responsive to a request for a resource which is not
7 available,

8 creating a block identifier based on the number of
9 threads temporarily denied the resource; and

10 blocking the thread using said block identifier;
11 and

12 responsive to a resource becoming available,

13 creating a run identifier based on the number of
14 threads that have been first forced to wait and
15 have been subsequently satisfied; and

16 running the thread using said run identifier.

8. A memory device for storing signals to structure the components of a digital computer to form a stationary queue for managing simultaneous access to scarce or serially re-usable resources by multiple process threads, comprising:

```
a sleep code routine for generating a unique block
identifier when a process thread temporarily cannot
gain access to said resource and must be suspended;
```

```
a wakeup code routine for generating a unique run
identifier when a next thread in line is to be
re-animated and granted access to said resource;
```

```
a wait counter for counting the cumulative number of
threads that have been temporarily denied the resource;
```

a satisfied counter for counting the cumulative number of threads that have been denied access and subsequently granted access to said resource;

said sleep code routine being responsive to said wait counter for generating said run identifier; and

said wakeup code routine being responsive to said
satisfied counter for generating said run identifier.